Discovery, Vol. 52, No. 246, June 1, 2016 RESEARCH

ISSN 2278–5469 EISSN 2278–5450

Discovery

Canonical Genetic Algorithm: Structure, Parameters, Operators and Adaptation Issues

Rohtash Dhiman[1], Saini JS[2], Priyanka[3]

1. Deptt. of Electrical Engineering, Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Sonipat, Haryana, India. Email: rohtash.k@gmail.com
2. Deptt. of Electrical Engineering, Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Sonipat, Haryana, India. Email: jssain@rediffmail.com
3. Deptt. of Electronics & Communicaton Engineering, Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Sonipat, Haryana, India. Email: priyankaiit@yahoo.co.in

Article History
Received: 18 April 2016
Accepted: 23 May 2016
Published: 1 June 2016

Citation
Rohtash Dhiman, Saini JS, Priyanka. Canonical Genetic Algorithm: Structure, Parameters, Operators and Adaptation Issues. *Discovery*, 2016, 52(246), 1497-1504

ABSTRACT

This paper puts forth a succinct account of Genetic Algorithms (GAs), their structure, parameters, operators and their adaptation issues. The description of implementation details of GA can be of great help even to a novice to code GA for any application at hand. Important applications, adaptation scenario and adaptation level in GA presented in this paper, forms an important survey. The paper intends to present some seminal/ landmark applications of GAs.

Key words: Genetic Algorithms (GAs), Probability of mutation (p_m), Probability of cross-over (p_c).

1. INTRODUCTION

Genetic algorithms (GAs) are search and optimization algorithms inspired by natural evolution of species [1]–[3]. Natural evolution is mainly based upon three operators, namely selection (which is fuelled on the premise of survival of the fittest but sparingly with a chance of growth for the week candidates too), reproduction and mutation of species. Every organism in nature has a set of rules or blueprints describing all the characteristics of the organism. These tiny building blocks of life are called genes of the organism, which are further connected to form long strings, named as chromosomes. Genes represent specific traits of the members of species, such as colour of eye or hair, height, sex, skin colour, etc. Genes and their attributes form organism's genotype, while behavioural expression of genotype is named as phenotype. In a canonical GA, an initial population of candidate solutions is generated by some random process and the evolution operators are used to modify the population of candidate solutions, the process is iterated till a stopping criterion is met. Mimicking of the process of natural evolution paved the way for use of this strong process of nature for engineering optimization and search. GAs initially proposed by John Holland in the 1960s were further contributed by De-Jong, Fogel, Michalewicz, Goldberg, Davis, Koza, Mitchell, Forrest, etc. [1]–[10]. In the present paper, a brief account GAs has been rendered along with the structure, parameters, operators & the adaptation issues. The applications and adaptation scenario in GAs has also been discussed.

2. STRUCTURE OF GENETIC ALGORITHM

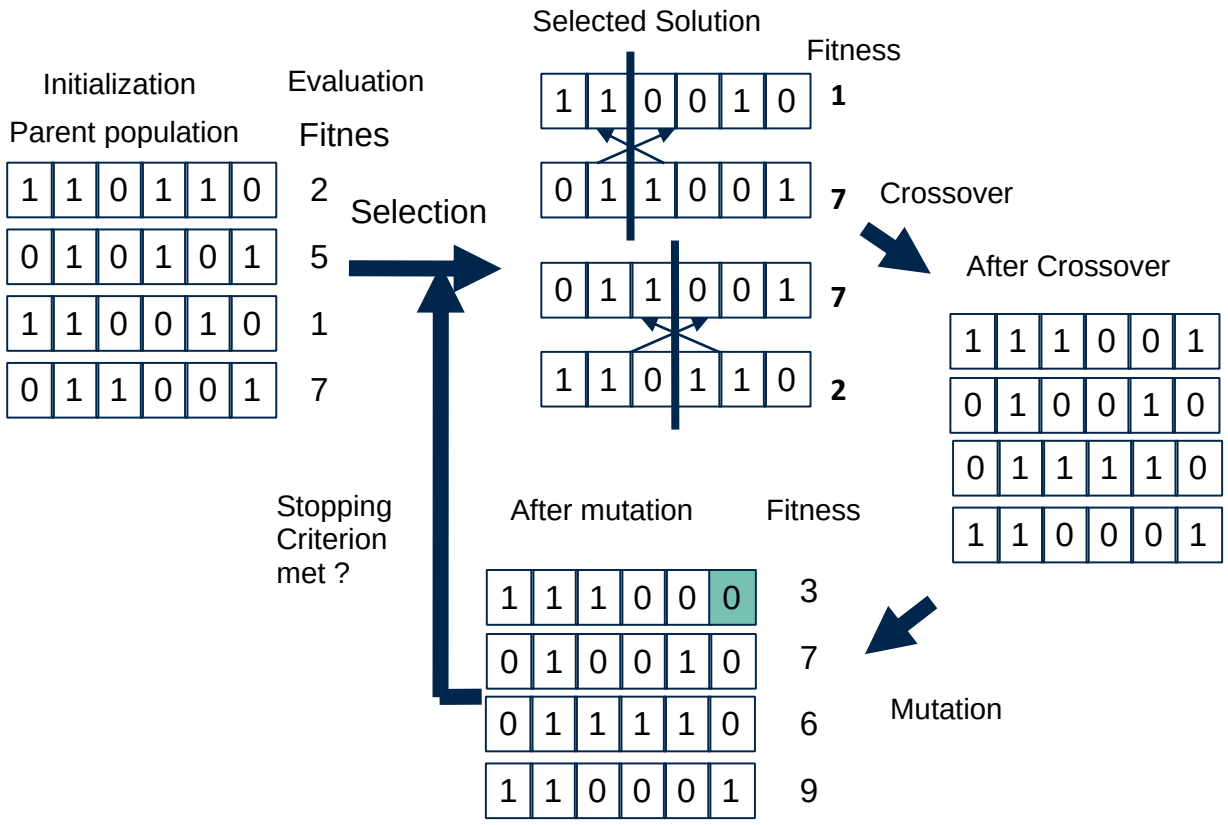


Figure 1 Structure of a binary coded Genetic Algorithm

Genetic Algorithm begins with a population of candidate solutions called chromosomes. Each chromosome contains building blocks known as genes. Each chromosome has some fitness, determined by a user-defined function, called the fitness function or objective function. The fitness function returns a value that is proportional to the chromosome's suitably to give an optimal solution to the problem at hand. The fitness function is problem- dependent. The GA may have the population of real numbers or binary numbers, thus defining the GA to be real-or binary-coded. Fig.1 depicts the structure of a binary coded GA. The algorithm starts with initialization of a population of candidate solutions drawn randomly and including a priori known good solutions. After initialization,

three operators namely selection, cross-over (reproduction) and mutation are used in a canonical GA. Elitism is another optional parameter which may be used whereby the best candidate to date is retained through the evolution/generations.

Selection of a chromosome for participation in cross-over (reproduction) is a random process but heavily biased by the chromosome's fitness. This interalia means that even weak candidates will stand some chance of selection. The chromosomes so selected form a mating pool for cross-over. The cross-over operator performs the mating of two chromosomes. One pair of chromosomes (parents) is randomly selected from the mating pool to participate into cross-over. The probability of cross-over (real number between zero and one) decides formation of two new chromosomes (children) from the parents. If allowed by the probability of cross-over, two child chromosomes are created which inherit complementing genetic material from their parents. The cross-over operator controls the passing of genetic material from each parent to the child chromosome. The new chromosomes produced after cross-over enter into the offspring pool for the next generation.

The mutation operator randomly changes part of the genetic make-up of a chromosome. Mutation rate is a user-defined parameter and decides the rate of occurrence of mutation on any chromosome.

New generation of chromosomes are thus formed by application of genetic operators (reproduction, cross-over and mutation) on the older population. The process of selection, cross-over, mutation and formation of a new population completes one iteration or generation of GA. A GA iterates, until certain stopping criteria (such as a fixed number of generations, or a time limit or a user-defined rule) is met. GAs are one type of Evolutionary Algorithms (EAs). The structure of a general Evolutionary Algorithm (EA) is given as [3]:

```
begin
        t: = 0;
initialize P (t);
evaluate P (t);
                while (not terminate) do,
                        t: = t+1;
P'(t): = select [P (t)];
P (t+1): = variation [P'(t) ];
Evaluate P (t+1);
                end;
end.
```

The algorithm maintains a population P(t) of 'n' chromosomes at generation 't'. Each chromosome represents a potential solution. Objective or fitness function is used to evaluate fitness of each chromosome. Then a new offspring population P(t+1) is formed at generation (t+1), by variation operators. A canonical Genetic Algorithm uses selection operator along with variation operators (i.e., cross-over and mutation).

3. PARAMETERS AND OPERATORS OF GENETIC ALGORITHM

GA starts with a randomly generated population of candidate solutions. However, if from prior knowledge of the problem domain, certain good solutions are known, they can also be incorporated into the initial population, giving a speedier evolution to optimal solution. This section describes the different operators and parameters of a canonical GA.

3.1. Initialization of Population

The representation used for initialization may be real or binary coded. The binary coding is explained in Fig.1. The algorithm for initialization of population may be given as [5]:

```
for u = 1:1: size of population
        Initialize variables to zero
        for t = 1:1:chromosome length
                Perform Coin flip using random number generator
                Store values in an array to construct Chromosome
         end;
            Store Chromosomes in a bigger array to form population
 end
```

This algorithm initializes a binary coded population with user-defined population size and chromosome length and stores the binary population in an array.

3.2 Selection

This operator mimics Darwinian principle of survival of the fittest. This operator selects chromosomes form present population to form a new population with probabilistic bias towards higher fitness of chromosomes. The chromosomes with higher fitness may get more copies in next generation while those with the lowest fitness may perish. So probabilistic biasing is such that the best chromosomes get more copies, the average stay even, and the worst die off. Many types of selection schemes are adopted in literature, some of them are: Roulette Wheel Selection (RWS), Stochastic Universal Sampling , Tournament Selection, Rank selection etc. The algorithm for RWS [5] is given below:

a) Calculate the fitness value fval (v_i) for each chromosome v_i (i = 1,2, ..., popsize). We assume that fitness values are positive, otherwise one can use some scaling mechanism to render them all positive.

b) Sum up above vector to find the total fitness of the population

F= $\sum$fval(v_i)

(c) Calculate the probability, p_i, of selection for each chromosome v_i.

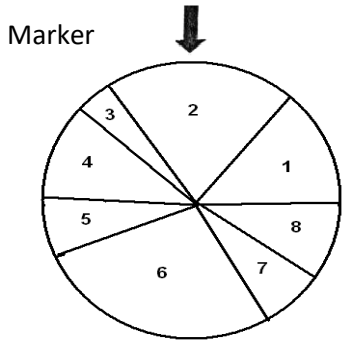
p_i = fval(v_i) / F; i = 1,2, ..., popsize.

(d) Calculate the cumulative probability, q_i, for each chromosome v_i (i = 1,2, ..., popsize), i.e.

$$q_i = \sum_{j=1}^{i} p_j$$

(e) Generate a real valued random number (float) in the range [0 to 1].

(f) If r < q_1, then select the first chromosome (v_i), otherwise select the i-th chromosome v_i *(2<i<popsize), such that* $q_{i-1}<r<<q_i$, Fig.2 shows RWS pictorially.



Slice Areas:
1: 14%
2: 23%
3: 03%
4: 11%
5: 06%
6: 25%
7: 08%
8: 10%
Total = 100%

Figure 2 Roulette Wheel Selection [7]

3.3. Cross-Over

The crossover operator picks successive pairs of chromosomes from the current population. The decision to perform cross-over or not on a chosen pair rests with probability of crossover p_c (also called crossover rate and is a user-defined parameter). Crossover rate is the probability

that two parents will undergo cross over. The crossover can either be single point or multi-point crossover. In the cases where no crossover allowed due to crossover rate being lower than a random number, then two offsprings formed are exact replicas (copies) of their respective parents.

Further, within the chosen pairs, a randomly chosen point (chosen with uniform probability) is selected as the crossing site. If, however, the crossing site does not fall between the first and the last bit (i.e. within the chromosome length) (as the crossing site is selected only randomly and not even a single such site may be allocated to a chromosome pair), then again the two off springs are exact copies of respective parents. However, if the randomly chosen locus (crossover site) is within the chromosome, then this crossover operator exchanges the subsequence before and after that locus (crossover site) between the two chromosomes to create two offspring as shown in Fig.3 for binary chromosomes [5].

Figure 3 One point Crossover on Binary Chromosomes of length six each

The algorithm given below may be used to perform Crossover [5]:

a) Generate a random number (float) 'r' in the range [0-1].
(b) If $r < p_c$, select the given chromosome pair for crossover.
(c) Generate a random integer number, *pos,* between 1 and L-l (where L= chromosome length). This number is crossover site.
(d) Two chosen chromosomes ($a_1a_2 \ldots a_{pos-1}a_{pos} \ldots a_L$) & ($b_1b_2 \ldots b_{pos-1}b_{pos} \ldots b_L$)
are replaced by a pair of their offspring:
($a_1a_2 \ldots\ldots a_{pos-1}b_{pos} \ldots b_L$) & ($b_1b_2 \ldots b_{pos-1}a_{pos} \ldots a_L$)

3.4. Mutation

This operator randomly (with probability of mutation, p_m, a user-supplied GA parameter also known as mutation rate) flips bits in a chromosome in case of a binary coded GA. Mutation is an occasional event with small probability, and randomly alters the allele value in a chromosome. In binary coding, mutation means flipping of bits from 0 to 1 and vice-versa. There are many mutation schemes used by researchers. Mutation can occur at each bit position in a string with probability, p_m (usually this probability is very small, e.g. 0.001 – 0.09). The following algorithm [5] can be used to perform mutation on binary chromosomes:

(a) For each chromosome in the current (i.e. after crossover) population and for each bit within the chromosome, generate a random number (float) 'r' in the range [0 to 1].
(b) If $r < p_m$, mutate the bit.

4. APPLICATIONS

GAs find applications in science, engineering, designing, pharmaceuticals, business, financial forecasting, and sociology; to narrate the whole story of applications shall be quite exhaustive, so the most noticeable contributions are outlined in this section. Nature-inspired computation regularly yields in amazing results while applying to complex search and optimization issues where conventional techniques are either not relevant or end up being unsuitable. GA is used to optimize parameters of digital filters [11], optimization of various design parameters of linear collider [12], refuelling of pressurized water reactors [13], optimization of aircraft design parameters [14][15], two dimensional shape representation [16], electromagnetic framework for portable mobile manipulators [17] optimization of configuration of survivable networks [8], and mixed integer evolution stratgies [18]. Standard GAs and their alternate forms are utilized to take care of combinatorial issues with straight forward representation of arrangements like production scheduling [19], set partitioning problem [20], vehicles routing [21], circuit switched telecommunication networks [22] and job shop scheduling [23]. GAs are also employed for issues identified with high specialization circuit design problems like planning of VLSI (Very Large Scale Integration) architectures and Electron Beam Lithography (EBL) [24], VLSI routing [25]. Genetic Programming is used for automated synthesis of electrical circuits [9]. An evolutionary approach for synthesis of high performance

analog integrated circuits is presented in [26]. In [27], Dengiz et al. present a local search GA for optimal design of reliable networks. The problem of channel asset administration is addressed in [28]. In [29], a steady state GA is used for generating fuzzy rules target tracking radar system. Xiao J. et al. [30], present planner/navigator for mobile robot. In reference [31], GA is used for automatic image registration by matching edge. The authors of [32] present an introduction to GA and evolution strategies. An accurate credit management system for business application is given in [33]. Genetic algorithm is used for design of a self learning fuzzy logic controller in [34]. The authors of [35] propose dynamic control of GA parameters using fuzzy logic. Hwang et al. [36] present the design of intelligent controller using GA. Design of membership function & rule sets for fuzzy logic controllers using GA is presented in [37]. GAs are used for optimal design of membership function & control rules simultaneously for FLCs [38]. A GA-based PID controller tuner, using a fitness function constructed as the inverse of Integral Square Error (ISE) is presented in [39]. GA based epileptic seizure detection scheme is presented in [40][41]. Image enhancement using GA presented in [42]. Further listing of applications and survey on GAs can be found in [43], [44].

5. ADAPTIVE SCENARIOS IN GAs:

The connection between GA control parameters settings and GA execution is considered as unpredictable relationship, yet there are approaches to comprehend this relationship [45] [46]. GAs tuned FLCs are can be found in [38][47][48], their fundamental idea had been to use an FLC whose inputs are any combination of GA execution measures or current control parameters, which yields upgraded control parameters as depicted in Fig.4 [49]. Adaptive GA using FLC is presented in [35] [50] and [51]. In [52], an adaptive GA based upon FLCs, is used for multi-target improvement issues. In [52] a fuzzy learning GA is proposed for uniform guess of pareto-ideal arrangements. Adaptive GA operators based on co-evolution with fuzzy behaviours presented in [53]. No free lunch theorem is presented in [54]. Optimization of Control Parameters for Genetic Algorithms presnted in [55]. The adaptation in GA is mainly reported by introducing tuning of GA parameters with some computational intelligence tool like FLCs, neural networks etc.

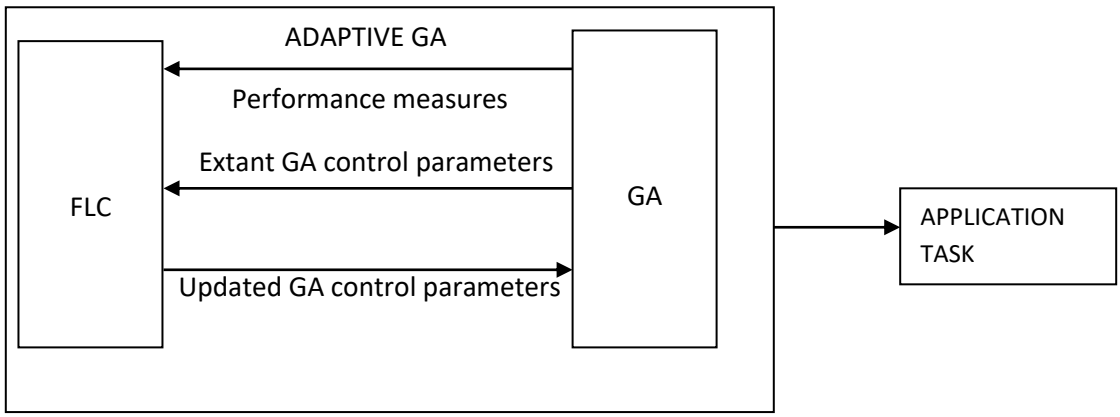


Figure 4 Structure of the AGA model based FLCs [49]

5.1. Adaptation levels in GAs:

The adaptation in Genetic Algorithms can be at three levels:

Population level adaptation: The control parameters applicable for entire population undergo adaptation. Different types of cross over & mutation operators are used & allotted an initial application probability. These probabilities are updated with the evolution process such that operators causing generation of better chromosomes are allotted higher probabilities & vice versa.

Individual level adaptation: Initially, each chromosome is allotted its own probability of cross over & mutation. Convergence state of the population & the fitness value of the chromosome decide the adaptive variation in the values of these probabilities in such a way that high fitness for the successive populations is achieved. [52].

Component level adjustment: These adjustments are powerful changes as to how the individual segment of every chromosome will be controlled freely from each other. The transformation probabilities connected with every piece of each chromosome are fused into a genetic representation, which develops with GA itself [30].

6. CONCLUSION

In the present paper, the structure of a simple Genetic Algorithm has been discussed. Various constituent operators of GA have been explained. The implementation algorithms have been given with details. However, the representation of population and that of objective function is dependent on the optimization problem at hand. The effective definition of the candidate solution and that of objective function make the GA work for the problem at hand. The adaptation level in GAs is touched in brief and several applications of GAs are also explained.

REFERENCES

1. J. H. Holland, "Adaptation in Natural & Artificial Systems," *Sgart Newsl.*, no. 53, p. 15, 1975.
2. J. H. Holland, "Outline for a Logical Theory of Adaptive Systems," *J. Assoc. Comput. Mach.*, vol. 03, pp. 297–314.
3. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1992.
4. K. A. De Jong, "An analysis of the behaviour of a class of genetic adaptive system," University of Michigan,Ann Arbor, 1975.
5. D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. T Addison-Wesley Publishing Company, Inc., 1989.
6. K. A. De Jong, "Are Genetic Algorithms Functions Optimizers?," in *parallel Problems Solving*, 1992.
7. L. J. Fogel, "Autonomous Automata," *Industrial Research Magazine*, pp. 14–19, 1962.
8. L. Davis, "A genetic algorithm for survivable network design," in *Proceedings of 5th iIternational conference on Genetic Algorithms*, 1993.
9. J. R. Koza, F. H. Bennett, D. Andre, M. A. Keane, and F. Dunlap, "Automated synthesis of analog electrical circuits by means of genetic programming," *IEEE Trans. Evol. Comput.*, vol. 1, no. 2, pp. 109–128, 1997.
10. S. Forrest and M. Mitchell, "What Makes a Problem Hard for a Genetic Algorithm? Some Anomalous Results and Their Explanation," *Mach. Learn.*, vol. 13, no. 2, pp. 285–319, 1993.
11. S. U. Ahmad, "Design of Digital Filters Using Genetic Algorithms," *Ph.D. Thesis, Univ. Victoria, Canada.*, 2008.
12. H. G. Beyer, "Some aspects of the evoluntionary strategy' for solving tsp-like optimization problems appearing at the design studies of a 0.5 TeV e+e- linear collider," in *parallel Problems Solving from nature 2, Amsterdam. Elsevier*, 1992.
13. T. Back, J. Heistermann, and C. Kappler, "Evolutionary algorithms support refueling of pressurized water reactors," in *Proceedings of IEEE International Conference on Evolutionary Computation*, 1996.
14. A. C. Marta, "Parametric Study of a Genetic Algorithm using a Aircraft Design Optimization Problem," Stanford University, Stanford, 2008.
15. J. Periaux, "Robust genetic algorithms for optimization problems in aero dynamic design," in *Genetic Agorithms in Engineering & Computer science*, 1995.
16. M. Schoenauer, "Shape representations for evolutionary optimization and identification in structural mechanics," in *Genetic Algorithms in Engineering and Computer Science*, Chichester: Wiley, 1995, pp. 443–463.
17. B. Anderson, "Configuration optimization of mobile manipulators with equality constraints using evolutionary programming," in *Proc. 1st Annu. Conf. on Evolutionary Programming. San Diego, CA: Evolutionary Programming Society*, 1992, pp. 71–79.
18. Schutz, "Applications of parallel mixed integer evolution strategies with mutation rate pooling," in *In Proc. 5th Annual conf. on Evolutionary Programming. Cambridge,MA: MIT press*, 1996, pp. 345–354.
19. R. Bruns, "Direct chromosome representation & advanced genetic operators for production scheduling," in *Proceedings of the 5th International Conference on Genetic Algorithms*, 1992, pp. 352–359.
20. D. M. Levine, "A genetic for the set partitioning problems," in *proc. 5th int. conf. on genetic algorithms, San Mateo,CA:Morgan Kaufmann*, 1993.
21. I. L. Blanron, "Multiple vehicle routing with time &capacity constraints using genetic algorithms," *Proc. 5th Int. Conf. Genet. Algorithms*, pp. 452–459, 1993.
22. L. A. Cox, Jr., L. Davis, and Y. Qiu, "Dynamic Anticipatory Routing In Circuit-Switched Telecommunications Networks," in *Handbook of Genetic Algorithms*, 1991, pp. 124–143.
23. M. T. Jensen, "Generating robust and flexible job shop schedules using genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 3, pp. 275–288, 2003.
24. F. Robin, A. Orzati, E. Moreno, O. J. Homan, and W. Bächtold, "Simulation and evolutionary optimization of electron-beam lithography with genetic and simplex-downhill algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 1, pp. 69–82, 2003.
25. J. Lienig, "A Parallel Genetic Algorithm for Performance-Driven VLSI Routing," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 29–39, 1997.
26. G. Alpaydin, S. Balkir, and G. Dundar, "An evolutionary approach to automatic synthesis of high-performance analog integrated circuits," *Evol. Comput. IEEE Trans.*, vol. 7, no. 3, pp. 240–252, 2003.
27. B. Dengiz, F. Altiparmak, and a. Smith, "Local search genetic algorithm for optimal design of reliable networks," *IEEE Trans. Evol. Comput.*, vol. 1, no. 3, pp. 179–188, 1997.
28. H. G. Sandalidis, P. P. Stavroulakis, and J. Rodriguez-Tellez, "An efficient evolutionary algorithm for channel resource management in cellular mobile systems," *IEEE Trans. Evol.*

Comput., vol. 2, no. 4, pp. 125–137, 1998.
29. K. C. C. Chan, V. Lee, and H. Leung, "Generating Fuzzy Rules for Target Tracking Using a Steady-State Genetic Algorithm," *IEEE Trans. Evol. Comput.*, vol. 1, no. 3, pp. 189–200, 1997.
30. J. Xiao and Z. Michalewicz, "Planner / Navigator for Mobile Robots," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 18–28, 1997.
31. J. Inglada and F. Adragna, "Automatic Multi-Sensor Image Registration by Edge Matching using Genetic Algorithms," in *Geoscience and Remote Sensing Symposium, 2001. IGARSS '01. IEEE 2001 International*, 2001, vol. 05, no. 0, pp. 2313 – 2315.
32. M. Dianati, I. Song, and M. Treiber, "An introduction to genetic algorithms and evolution strategies," *Univ. Waterloo, Canada*, 2002.
33. A. Tsakonas, N. Ampazis, and G. Dounias, "Towards a comprehensible and accurate credit management model: Application of four computational intelligence methodologies," in *Proceedings of the 2006 International Symposium on Evolving Fuzzy Systems, EFS'06*, 2006, pp. 295–299.
34. C.-K. Chiang, H.-Y. Chung, and J.-J. Lin, "A Self-Learning Fuzzy Logic Controller Using Genetic Algorithms with Reinforcements," *IEEE Trans. Fuzzy Syst.*, vol. 5, no. 3, pp. 460–467, 1997.
35. M. A. Lee and H. Takagi, "Dynamic Control of Genetic Algorithms using Fuzzy Logic Techniques," in *5th Int'l Conf. on Genetic Algorithms (ICGA'93), Urbana-Champaign, IL, July 17-21, 1993*, pp. 76–83.
36. W. R. Hawang, "An Intelligent controller design based on genetic algorithms," in *Proc. of 32nd conf. on decision & control, San Antonio, TX, 15-17 December*, 1993, pp. 1266 – 1267.
37. A. Homaifar and E. McCormick, "Simultaneous Design of Membership Functions and Rule Sets for Fuzzy," *IEEE Trans. Fuzzy Syst.*, vol. 3, no. 2, 1995.
38. J. S. Saini, M. Gopal, and A. P. Mittal, "Evolving Optimal Fuzzy Logic Controllers by Genetic Algorithms," *IETE J. Reasearch*, vol. 50, no. 3, pp. 179–190, 2004.
39. J. S. Saini, M. Gopal, and A. P. Mittal, "Genetic Algorithm Based PID Tuner," *J. Institutuin Eng. Electr. Eng. Div.*, vol. 85, pp. 216–221, 2005.
40. R. Dhiman, J. S. Saini, and Priyanka, "Genetic algorithms tuned expert model for detection of epileptic seizures from EEG signatures," *Appl. Soft Comput. J.*, vol. 19, pp. 8–17, 2014.
41. H. Ocak, "Optimal classification of epileptic seizures in EEG using wavelet analysis and genetic algorithm," *Signal Processing*, vol. 88, pp. 1858–1867, 2008.
42. M. Paulinas and A. Ušinskas, "A survey of genetic algorithms applications for image enhancement and segmentation," *Inf. Technol. Control*, vol. 36, no. 3, pp. 278–284, 2007.
43. T. Back, U. Hammel, and H.-P. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 3–17, 1997.
44. A. Munawar, M. Wahib, M. Munetomo, and K. Akama, "A survey: Genetic algorithms and the fast evolving world of parallel computing," *Proc. - 10th IEEE Int. Conf. High Perform. Comput. Commun. HPCC 2008*, pp. 897–902, 2008.
45. D. Reynolds and J. Gomatam, "Stochastic modeling of genetic algorithms," in *Artificial Intelligence*, pp. 303–330.
46. O. François and C. Lavergne, "Design of evolutionary algorithms - A statistical perspective," *IEEE Trans. Evol. Comput.*, vol. 5, no. 2, pp. 129–148, 2001.
47. F. Herrera and M. Lozano, "Adaptation of Genetic Algorithm Parameters Based on Fuzzy Logic Controllers," *Genet. Algorithms Soft Comput.*, vol. June, pp. 95–125, 1996.
48. M. A. Lee, "A framework for studying the effects of dynamic crossover,mutation,& population sizing in genetic algorithms," in *in Advances in Fuzzy Logic,Neural Network &Genetic Algorithms. Germany: Springer-Verlag*, pp. 11–126.
49. J. S. Saini, M. Gopal, and A. P. Mittal, "Adaptive Evolutionary Tuner for Adaptive Fuzzy Logic Controller," in *IEEE-IAS &PES sponsored "International Conference"Nov 9-10,2004, New Delhi*.
50. F. Herrera and M. Lozano, "Fuzzy Evolutionary Algorithms and Genetic Fuzzy Systems: A Positive Collaboration between Evolutionary Algorithms and Fuzzy," *Artif. Intell.*, pp. 83–130, 2009.
51. Y. Shi, S. Member, R. Eberhart, and Y. Chen, "Implementation Of Evolutionary Fuzzy Systems - Fuzzy Systems," *IEEE Trans. Fuzzy Syst.*, vol. 7, no. 2, pp. 109–119, 1999.
52. S. Voget and M. Kolonko, "Multidimensional Optimization with a {F}uzzy {G}enetic {A}lgorithm," *J. Heuristics*, vol. 4, no. 3, pp. 221–244, 1998.
53. F. Herrera and M. Lozano, "Adaptive genetic operators based on coevolution with fuzzy behaviors," *IEEE Trans. Evol. Comput.*, vol. 5, no. 2, pp. 149–165, 2001.
54. D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, 1997.
55. J. Grefenstette, "Optimization of Control Parameters for Genetic Algorithms," *IEEE Trans. Syst. Man. Cybern.*, vol. 16, no. 1, pp. 122–128, 1986.